IP2

ISAAC Petascale Image Processing

Getting Started

v0.05

1 Introduction

1.1 What is IP2?

The shortest answer is ISAAC Petascale Image Processing (IP2), which hints at a few things, but is otherwise not terribly informative. IP2 is an astronomy image processing application designed to support a very effective, but very computationally intensive, image differencing method. The image processing in IP2 uses commodity parallel-processing methods for significant acceleration. While the original application is primarily image differencing (a.k.a. subtraction), this differencing function is a first utility in what is intended to be a more general pipeline application for high-speed processing of large astronomical images.

To address the scalability and performance required by large data volumes, the current data acquisition, processing and analyses algorithms require review, and in some cases, rewrite. Several efforts are underway to attain the needed high-performance computing by exploiting the emerging hardware availability, and development software support, of massively parallel many-core and accelerator architectures. In collaboration with one such effort spearheaded at UC Berkeley and Lawrence Berkeley National Laboratory (LBNL), titled Infrastructure for Astrophysics Applications Computing (ISAAC), IP2 was initially created to explore acceleration of a non-traditional and high-impact spatially-varying convolution algorithm as a part of astronomical image subtraction.

The initial focus of this application has been accelerating a computationally intensive method for image differencing in astronomy. The technique is known as Optimal Image Subtraction (OIS) [1] which uses a convolution technique to match the point spread function (PSF) between images. In many situations, especially with larger images, the PSF can vary across the image, requiring a spatially-varying convolution [2] in order to achieve high quality subtractions.

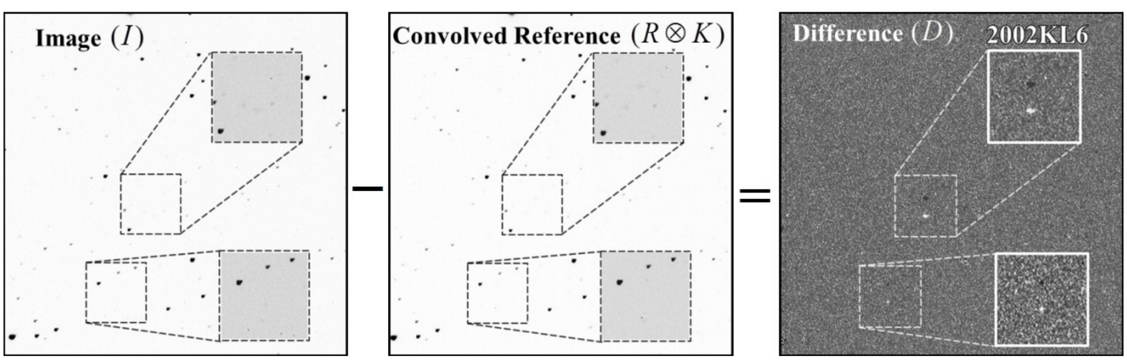

Figure 1.1 Example OIS subtraction. Two images taken at different times are matched and subtracted, yielding a difference image showing what has changed photometrically. The difference image on the right reveals two faint asteroids that have moved in the time between exposures. Image credit for the original exposures on the left: NEAT, courtesy NASA/JPL-Caltech. The difference image on the right has been generated with IP2.

The convolution in OIS relies on fitting a superposition of basis functions to describe the convolution kernel. Traditionally this has been done using a superposition of Gaussian function bases (GFB). However, the GFB assumes a highly symmetric PSF of a Gaussian nature. The PSF from an optical telescope should tend toward a superposition of Gaussian functions in the ideal situation. In reality, many images have asymmetric and non-Gaussian PSFs. This can be caused by any number of effects, from atmospherics and the telescope itself to side effects from adaptive optics or other upstream image processing. The ideal basis is the Dirac delta function basis (DFB) [3], but the DFB adds dramatically to the computation (typically an order of magnitude, but up to two orders). Using OpenMP or GPUs can dramatically accelerate the DFB OIS [4], making it practical for applying to large images with high acquisition cadence or very large data sets. The acceleration can be over three orders of magnitude from the IDL implementation of the 2nd-order fit DFB OIS for large images, as shown in Figure 1.2.

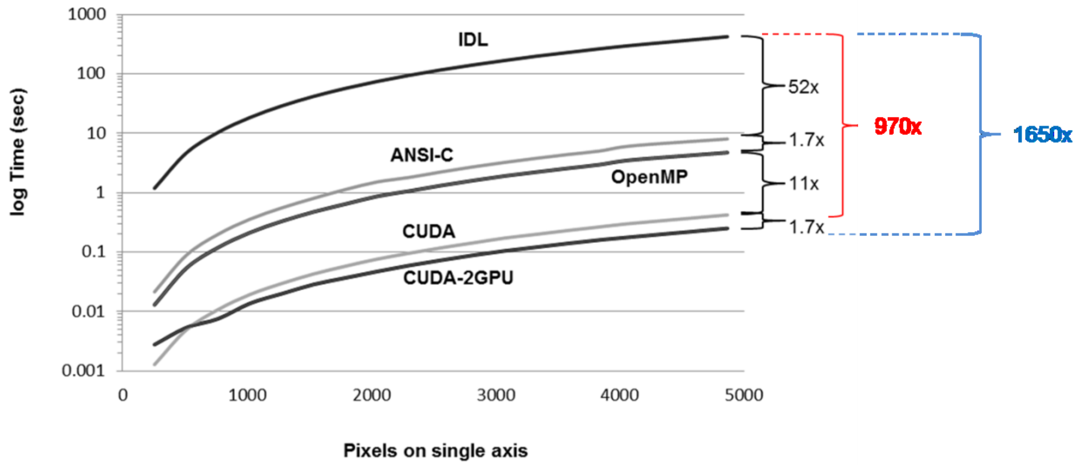

Figure 1.2 Computation times for the DFB spatially-varying convolution using the original IDL starting point compared to the IP2 accelerated CPU and GPU implementations.

For smaller images, IP2 offers little to no advantage over traditional single threaded code, but images continue to grow in size, and even the tiles of mosaic cameras are now usually larger than 2k x 2k. As CCD technology continues to advance, raw images and tiles on the scale of 10k x 10k are on the immediate horizon.

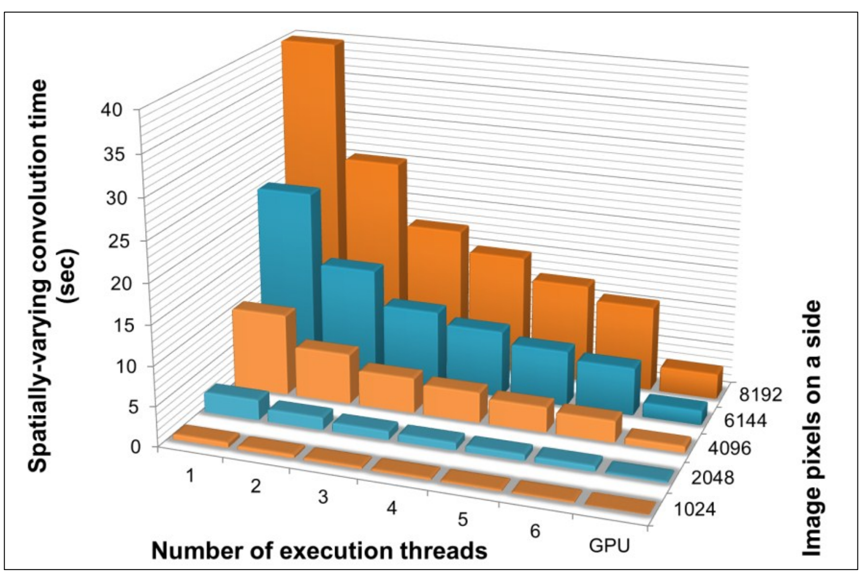

Figure 1.3 Dirac delta function spatially-varying convolution speeds using multi-core CPUs or a GPU with IP2.

IP2 also introduces a new technique for ameliorating a tendency of the DFB OIS to over-fit the kernel to noise in the image. A PSF cross-correlation (PCC) method is applied prior to the kernel fitting process [5]. The method can be highly effective at improving signal to noise (SNR) when searching for point source objects in the difference images. In some circumstances it can lead to increased false detections due to transients such as cosmic ray strikes, if cosmic rays have not been removed by other means in advance, and it is often not suitable for extended source detections such as light echoes or variable nebula.

1.2 Installation

This is an Alpha release of IP2. It operates on a single node computer. A cluster-scalable version has been demonstrated and will be available in a public release in the near future.

1.2.1 Hardware and Software Requirements

The initial Alpha level release of IP2 is only supported under 64-bit Linux. The application requires that the gcc and g++ compilers are installed for proper linking of the IP2 binaries to the local libraries. Additionally, the GOMP library for the GNU version of OpenMP multi-core CPU support must be installed. There are two versions of the application, one with GPU support, and one without. The development and testing has been primarily done under Ubuntu 10.04 LTS, but this should not be a requirement for proper operation. Additional testing has taken place in HPC environments using other Linux distributions as well.

For the GPU supported version, CUDA 4.1 or higher must be installed. Only NVIDIA GPU cards with a compute capability 2.0 or higher will be used for GPU computation in IP2.

In the Alpha version of the code, Interactive Data Language (IDL) must be installed for the DFB or PCC versions of image subtraction. If no IDL is installed, IP2 will use the Becker GFB implementation of OIS[1], without acceleration. Future versions will remove the IDL dependencies and potentially add acceleration to the Becker implementation.

1.2.2 Installing IP2

Copy the downloaded tar file to the desired location and extract the contents, e.g.:

```
tar -xzf ip2sn_Linux64.tgz
```

Then link the code to the local library's:

```
cd ip2sn
./buildip2sn.sh
```

If your GPU libraries are in a non-standard location (/usr/local/cuda has been assumed), you may need to edit the script `buildip2sn.sh` and supply the correct paths (clearly marked near line 48). For non-GPU builds this is not an issue.

[1] http://www.astro.washington.edu/users/becker/hotpants.html

The build script will attempt to automatically detect the presence of CUDA and will build the executable `ip2sn_gpu` if it is able to. If CUDA cannot be identified, it will generate the CPU-only version of the executable simply called `ip2sn`.

If successful, the build process places the executable `ip2sn` or `ip2sn_gpu` into the ip2sn directory that the build script was launched in. If you move the executable to another location, you must move the idl directory with it. At this time the executable must be run from the directory in which it is installed for the linkage to the IDL support code. This inconvenience will be eliminated in the next release.

For OpenMP, the environment variable `OMP_NUM_THREADS` should be set to the number of available CPU cores (or possibly the total number of CPU cores minus one on some systems) for the best performance.

1.3 Running the Basic Test Sample

The distribution includes some simple prebuilt examples that can be used to verify the operation, and serve as samples for creating your own processing operations. IP2 uses a plain text input file called a recipe to describe what is to be done. The use of a recipe avoids the need for an overly complex command line. To run the application only requires the application name followed by a relative or absolute path to the recipe file, e.g.:

```
ip2sn_gpu myrecipe.ip2
```

To verify the installation run:

```
./ip2sn_gpu testdata/shortrecipe.ip2
or
./ip2sn testdata/shortrecipe.ip2
```

depending on the installation. The short recipe just issues a shell script command ls for the current directory and places the output in file called shelltest.txt if it is successful. In the GPU version it will also scan for available supported GPUs and print messages about what it finds.

An actual subtraction test can be run by executing:

```
./ip2sn_gpu testdata/test512ois.ip2
or
./ip2sn testdata/test512ois.ip2
```

depending on the installation. This will subtract two small 512x512 pixel images and place the output image into the directory testdata/results. You are encouraged to use this test script as a template for setting up your own OIS subtractions. Additional recipe file details are described in the next section.

2 Recipe Files

2.1 Command Structure

Commands are defined in a recipe file and structured within a task or a series of tasks. The concept of the task is to encapsulate operations that require multiple commands, but that are operating on the same set of image files. An example of such a scenario for task encapsulation is a pair of images that must be calibrated and then co-added.

2.1.1 Recipe Files

Tasks and commands are defined in a plain text recipe file. The performance motivation of the recipe file is to eliminate the overhead and complexity of writing complex shell scripts in order to describe a large number of image processing steps. Simple helper applications or shell scripts can be used to automate the generation of the recipe files, or they can be written by hand in any text editor. The basic structure of the recipe is that of a keyword-parameter pair on a single line:

```
[keyword] <parameter string> [end of line].
```

Note that keywords and parameters are case sensitive.

The purpose of the recipe file is to provide a human readable and writable interface to IP2. Once the application is launched, the recipe file is immediately parsed and converted into linked-lists of data structures for processing in the compute nodes. The parsing is accomplished by high-speed matching of keywords in the recipe file to a lookup table. Once matched, the remainder of the line is processed as a parameter, where the parameter type is also specified in a lookup table. Any necessary conversions from a text string variable to integers or floating point numbers take place at this stage. Keywords and parameters are converted into data structures that are linked in the order of discovery, which, for command sequences, becomes the order of execution.

The syntax is that of nested `Task` and `Cmd` (command) operations. All command parameters must appear between the `Cmd` keyword and an associated `CmdEnd` keyword. All commands must be fully contained within a task between the `Task` and `TaskEnd` keywords. The basic structure of the task/command relationship is illustrated in Figure 2.1 and demonstrated in the example recipe shown in Figure 2.6.

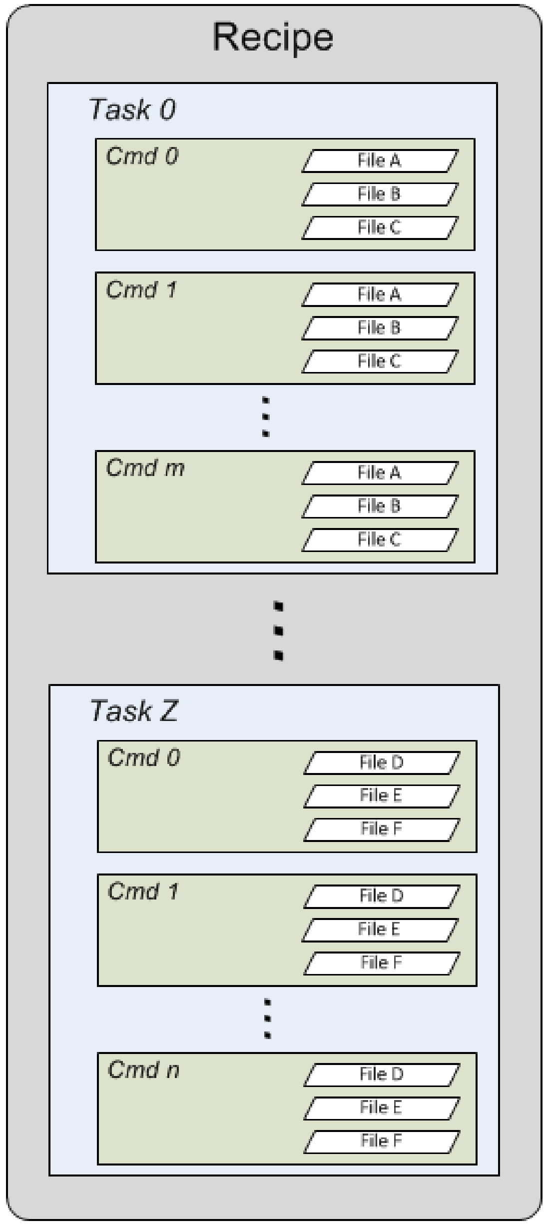

Figure 2.1 Diagram of task and command hierarchical structure

In the example recipe that follows in Figure 2.6, there are two independent tasks, each operating on separate image files. The tasks can be processed on the same node, each in turn, or they can be sent to separate nodes for simultaneous processing. The first task in the example performs a calibration step on a set of images, followed by a co-addition of the calibrated results. The second task performs an OIS subtraction of two other images.

2.2 Image Manipulation Commands

Images inputs for OIS must be in the standard FITS file format [6], used widely throughout astronomy. FITS file images must not be in compressed format.

The primary motivation for the creation of IP2 is to provide an accelerated platform for OIS. But a single operation tool, even one as powerful as OIS, is of limited appeal. A full-featured pipeline has many image manipulation operations to perform. The most basic operations are those used in image calibration. In this context calibration is the process taken to clean up the image data from the raw form that comes directly off of the detector, a CCD in most cases, by removing instrument induced artifacts. Several basic image math operations have been implemented in IP2 to support standard calibration functions. The standard calibration operations in Table 2-1 are all implemented in OpenMP parallelism. Because most of the commands only perform one or two mathematical operations per pixel on the image data, the computation time is not sufficient to cover the data movement overhead required for GPU implementations, thus GPU implementations are actually slower than OpenMP for these operations.

Table 2-1 IP2 Basic Commands for Image Calibration

Command	Description
Add	Basic pixel by pixel addition of two or more images
Median Add	Adding two or more images while maintaining scale, this is useful for creating co-added or “stacked” images that are scaled to the level of the initial input images
Subtract	Basic pixel by pixel subtraction (not OIS), useful for subtracting dark and bias images to remove amplifier and thermal noise
Scale	Multiply all pixels in an image by a given floating point value
Multiply	Multiply two images pixel by pixel
Flat Field	A scaled divide of an image by a flat field image, see below for details
Job Level Options	Description
ConfFlatSamp	Flat-field sample region size in pixels on a side. Default is 50, meaning a sample 50x50 pixel region in the center of the flat-field image is used to determine is the scaling offset.

Table 2-2 lists the OIS commands currently implemented in IP2. These commands currently rely on a hybrid implementation of a modified Miller code suite in IDL and CUDA acceleration of the spatially-varying convolution. Alternatively an OpenMP version will be used for the acceleration of the spatially-varying convolution in the event that no GPU is found or if the GPU operation is suppressed through the ConfNoGPU option in the recipe file.

Images must be co-aligned in an external tool prior to OIS subtraction in this release of IP2. Images up to 8k x 8k pixels have been tested successfully on the GPU version, and larger images

Table 2-2 IP2 OIS Commands

Command	Description
OIS	Complete second-order DFB OIS between two images. The inputs are the image and the reference image to be subtracted. The output is the OIS subtraction result image. This code is currently implemented in IDL.
Job Level Options	**Description**
ConfNoGPU	0 = default, GPU will be used if found, 1 = suppress GPU usage (CPU only) even if a suitable GPU is found
ConfImageSat	Pixel value for saturation (should actually be set for the level above which pixel response may be non-linear). Default value is 50000
ConfUseGauss	0 = default, OIS will use a 2nd-order Miller Dirac delta function basis (DFB) for the spatially-varying convolution kernel. 1 = use the traditional Gaussian function basis (GFB) for the spatially-varying convolution kernel implemented through the Becker HOTPanTS subtraction.
ConfUsePCC	0 = default, standard DFB. 1 = use PSF cross-correlation method for DFB noise reduced subtraction. This option has no effect on GFB subtraction.

2.2.1 Basic Calibration Commands

Raw data as read directly from a CCD has several sources of detector induced and optically induced artifacts. Systemic thermal photons and offsets induced by digital to analog converters and amplifiers from the instrument and the environment can be reduced through the use of dark frames and bias frames [7]. Dark frames are images made from exposures with a cover over the telescope or the shutter on the camera closed, an example is shown in Figure 2.2. The dark frame is taken for the same exposure duration and at the same detector temperature as the science images. As a result, the dark frame is a “picture” of the thermal noise photon characteristics of the detector. A second type of dark frame is the bias frame, an example of which appears in Figure 2.3. The bias frame has zero exposure time, and thus it characterizes an image of the offsets induced by the electronics in the camera readout stage. Once captured, the bias and the dark can both be subtracted from a raw image through a simple pixel-by-pixel subtraction. It is standard practice to co-add several samples of dark and bias frames into a master dark and master bias in order to reduce random noise effects. The very simple operation for dark frame subtraction is shown in equation(0.1).

$$Corrected_{x,y} = Original_{x,y} - \left(\frac{\sum_N Dark}{N} \right) - \left(\frac{\sum_M Bias}{M} \right) \qquad (0.1)$$

While the dark frames primarily correct for detector issues, the flat frames correct for optical defects in the telescope system [7]. The flat field image can capture issues with even small amounts of dust and debris on any of the optical surfaces, including mirrors, filters, or lenses. The flat field image is also able to characterize uneven illumination effects across the FOV such as those induced by vinetting. The flat field image is acquired by taking an exposure of an even “flat-light” illuminated field, hence the name. It is also standard practice to co-add several samples of the flat images into a master flat in order to reduce random noise effects. An example flat field image is shown in Figure 2.4.

To remove the flat field artifacts from the science image, it is necessary to divide the image by a normalized version of the flat field image. The normalization factor is acquired by finding an average pixel value from a sample region near the center of the flat field, were darkening effects due to vinetting are minimal. The science image is then divided, pixel by pixel, by the flat field.

An example of basic image processing is illustrated in Figure 2.5. Actual processing results will vary greatly depending on the calibration images. At this time, IP2 does not represent any innovation in basic image calibration commands, they are merely provided so that the pipeline can keep data internal to the system while preparing images for subtraction or future IP2 features. Future work may examine refinements to these basic image manipulation and calibration operations.

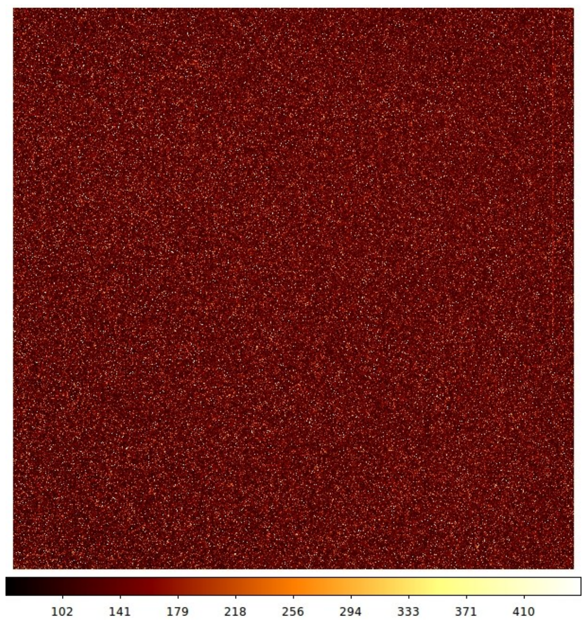

Figure 2.2 Example stacked master dark frame. A stacked image of 8 individual dark combined using IP2. Image credit: W. Green, F. Mezzalira and S. Hartung/SBO.

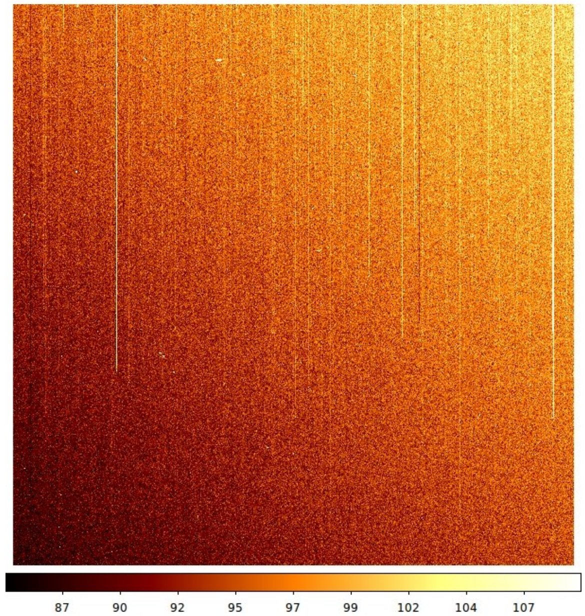

Figure 2.3 Example stacked master bias frame. A stacked image of 10 individual bias frames combined using IP2. Image credit: W. Green, F. Mezzalira and S. Hartung/SBO.

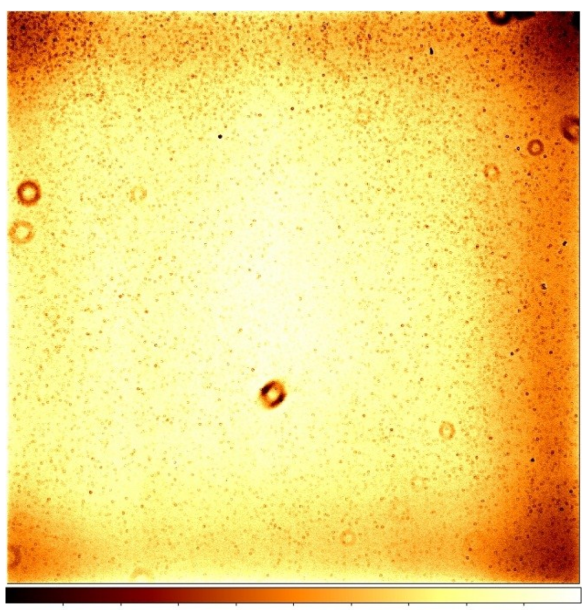

Figure 2.4 Example stacked master flat frame. Artifacts are due to dust and debris on the optics, which are out of focus and generate circular Airy disk patterns, and vignetting seen as a darkening in the corners. Image credit: W. Green, F. Mezzalira and S. Hartung/SBO.

$$MasterFlat_{x,y} = \frac{\sum_{N} Flat_{x,y}}{N}$$

$$Corrected_{x,y} = \frac{Original_{x,y}}{MasterFlat_{x,y}} MasterFlat_{SampleAverage} \tag{0.2}$$

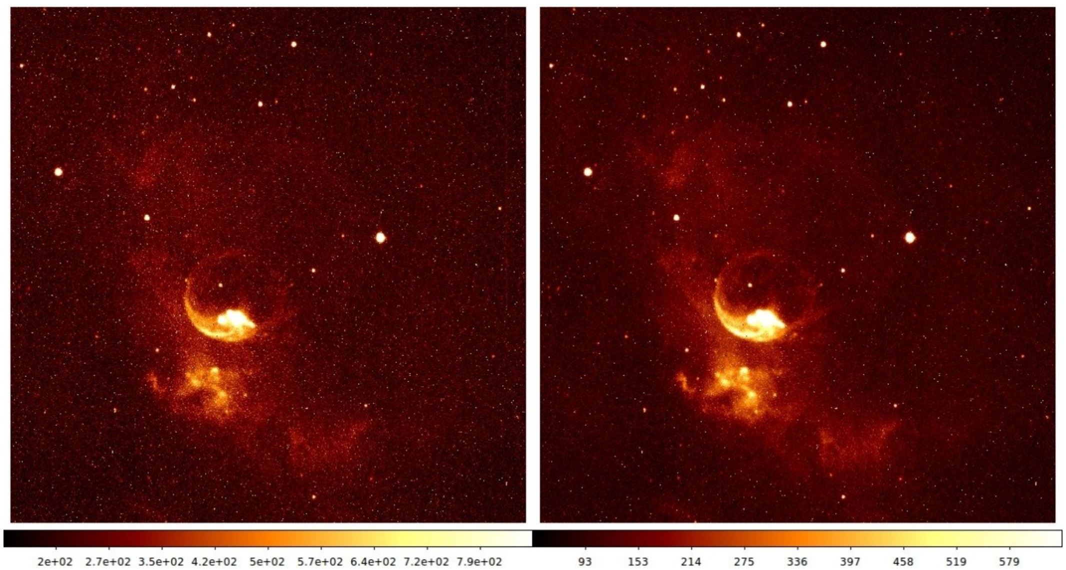

Figure 2.5 Example raw image on the left as read from the CCD, image dark subtracted and flat fielded in IP2 on the right. While some random noise remains, a great deal of the systemic artifacts has been removed. False color image of the Bubble nebula in hydrogen-alpha. Image credit: W. Green, F. Mezzalira and S. Hartung/SBO.

2.3 Diagnostic and Utility Commands

It is convenient to use the recipe command interface to provide other functions in addition to the built-in image manipulation routines. The extended diagnostic and utility commands are listed in Table 2-3.

The shell command, in particular, allows for a great deal of flexibility by allowing an arbitrary string to be passed to the operating system command line. Simple uses of the shell allowing recipe files to make directories or move files as needed. The shell command also allows the use of externally compiled tools (e.g. the Astromatic application suite) for accessing functionality that is not yet available inside of IP2.

Table 2-3 IP2 Diagnostic and Utility Commands

Command	Description
ImgStats	Performs a basic analysis on a specified image to determine max, min, and standard deviation characteristics of the image as a whole and of the sky background
Shell	Allows passing a command string to the operating system command shell for execution from a recipe file

Figure 2.6 shows an example IP2 recipe file.

```
# Input file for the IP2 pipeline defining the processing recipe
# This is a comment
JobName [some job name]
User [some user name for logging]
# for OIS set the option to use the PCC method
ConfUsePCC 1
# set the image pixel value of the saturation level
ConfImageSat 60000

Task NameForTask1
   # the common base directory for all tasks
   JobBaseDir /home/worker/
# dark subtract images
   # basic subtraction works for calibration and de-trending
   Cmd Subtract
          # relative sub-directories and file names
          InputPath swtest/sbo/bubble/
          OutputPath swtest/sbo/bubble/
          OutputFitsFile CAL_00000200.Bubble_Nebula.FIT
          InputFitsFile SCI_00000200.Bubble_Nebula.FIT
          InputFitsFile DARK_-20_Ha.FIT
   CmdEnd
   Cmd Subtract
          InputPath swtest/sbo/bubble/
          OutputPath swtest/sbo/bubble/coadd/
          OutputFitsFile CAL_00000207.Bubble_Nebula.FIT
          InputFitsFile SCI_00000207.Bubble_Nebula.FIT
          InputFitsFile DARK_-20_Ha.FIT
   CmdEnd
# median combine the calibrated images
   Cmd MedianAdd
          InputFileCount 3
          InputPath swtest/sbo/bubble/
          OutputPath swtest/
          OutputFitsFile medadd_img.fits
          InputFitsFile CAL_00000200.Bubble_Nebula.FIT
          InputFitsFile CAL_00000207.Bubble_Nebula.FIT
   CmdEnd
TaskEnd

# define another task that operates on a different set of files
Task NameForTask2
      # perform Optimal Image Subtraction (OIS)
      Cmd OIS
          InputPath tmt/
          OutputPath tmt/diff/
          OutputFitsFile A-B_diff.fits
          InputFitsFile A_img.fits
          InputFitsFile B_img.fits
      CmdEnd
TaskEnd
```

Figure 2.6 Example IP2 recipe file with two independent tasks

[1] C. Alard and R. H. Lupton, "A Method for Optimal Image Subtraction," *The Astrophysical Journal,* p. 325, 1998.
[2] C. Alard, "Image subtraction using a space-varying kernel," *Astronomy & Astrophysics Supplement Series,* vol. 144, pp. 363-370, 2000.
[3] J. P. Miller*, et al.*, "Optimal Image Subtraction Method: Summary Derivations, Applications, and Publicly Shared Application Using IDL," *Publications of the Astronomical Society of the Pacific,* vol. 120, pp. 449-464, 2008.
[4] S. Hartung*, et al.*, "GPU Acceleration of Image Convolution using Spatially-varying Kernel," presented at the IEEE International Conference on Image Processing (ICIP) 2012, Orlando, Florida, 2012.
[5] S. Hartung. (2013, January 1, 2013). Image Subtraction Noise Reduction Using Point Spread Function Cross-correlation. *ArXiv e-prints 1301,* 1413. Available: http://adsabs.harvard.edu/abs/2013arXiv1301.1413H
[6] R. J. Hanisch*, et al.*, "Definition of the Flexible Image Transport System (FITS)," *Astronomy & Astrophysics,* vol. 376, pp. 359-380, 2001.
[7] S. B. Howell, *Handbook of CCD Astronomy*, 2nd ed. vol. 5. New York, NY: Cambridge University Press, 2006.